

# **CropWatch-RDSIF**

Real-time delivery of spatial-temporal information for farmers.

## **User Manual (document)**

**UNE COSC591/594 (Group A & C)**

COSC594 students:  
Jason O. Aboh  
Steve Daniel

COSC591 students:  
Zoe Dowsett  
Qiaoling (Ling) Chen

## **Table of contents**

<b>Project team and Client</b>	<b>2</b>
<b>Stakeholders</b>	<b>2</b>
<b>User Manual</b>	<b>3 - 48</b>
<b>CropWatch-RDSIF: Cloud Function to obtain farm-calculations data</b>	<b>7 - 25</b>
<b>CropWatch-RDSIF: Exporting farm data to the Database (BigQuery)</b>	<b>26 - 30</b>
<b>CropWatch-RDSIF: Visualizing Database information in DataStudio</b>	<b>31 - 42</b>
<b>Repository (Github link)</b>	<b>42</b>
<b>Notices</b>	<b>43</b>
<b>Licenses</b>	<b>44 - 48</b>

## **Project:**

### **CropWatch: Real-time Delivery of Spatial-temporal Information for Farmers.**

CropWatch-RDSIF is an application built to deliver interactive maps to farmers, showing details of their crops over time. The data for the maps are time-series data obtained from satellite and other spatio-temporal sources. The application gathers the data, filters out important information bands within the data, then exports the data into a cloud database which conducts transformations and calculations on that data to provide useful information to farmers via data visualisation reports.

## **Project team and client**

### **UNE COSC591 students:**

- Zoe Dowsett
- Qiaoling (Ling) Chen

### **UNE COSC594 students:**

- Jason O. Aboh
- Steve Daniel

### **Client:**

- James Brinkhoff from Applied Agricultural Remote Sensing Centre (AARSC)

## **Stakeholders**

- Project team: Zoe Dowsett, Qiaoling (Ling) Chen, Jason O. Aboh, and Steve Daniel
- Client: James Brinkhoff (AARSC)
- Project sponsor/Supervisor: Edmund Sadgrove (UNE)
- Suppliers: Google Cloud Platform, satellite and other spatio-temporal sources.
- End users: Farmers in Australia

CropWatch

Real-time delivery of spatial-temporal information for  
farmers

# **User Manual**

[page left blank on purpose]

CropWatch-RDSIF

*Real-time delivery of spatial-temporal information for farmers*

*User manual*

*By: Jason O. Aboh*

*Release 0.1*

Before using this information, please be sure to read the general information contained under the “Notices” on page 42 and “Licenses” chapter on page 43 to page 47.

This edition applies to

©Copyright 2022 Jason O. Aboh, Steve Daniel, Zoe Dowsett, Qiaoling (Ling) Chen  
©Copyright 2022 James Brinkhoff

Licensed under the Apache License, Version 2.0 (the "License");  
you may not use this file except in compliance with the License.  
You may obtain a copy of the License at  
<http://www.apache.org/licenses/LICENSE-2.0>

## CropWatch-RDSIF: Cloud Function to obtain farm-calculations data

The python code to create the CropWatch-RDSIF data is hosted within a Google Cloud Platform function using a Python 3.9 runtime. The function is named ‘farm-calculator’ and consists of 4 files: “main.py”, “requirements.txt”, “farm\_details.json” , and “sa-private-key.json. These files are required to run the cloud function successfully.

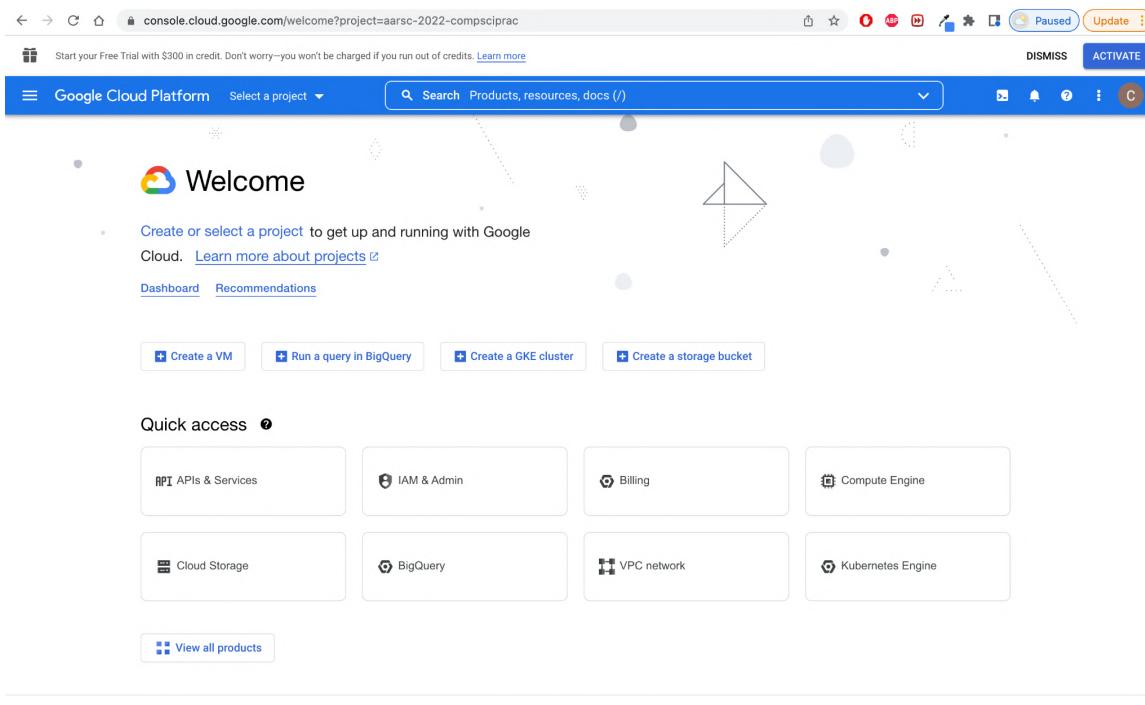
### STEPS:

#### Getting to the project:

- After project set up and an appropriate assignment of permissions and access to team members; Open up your browser and head to :

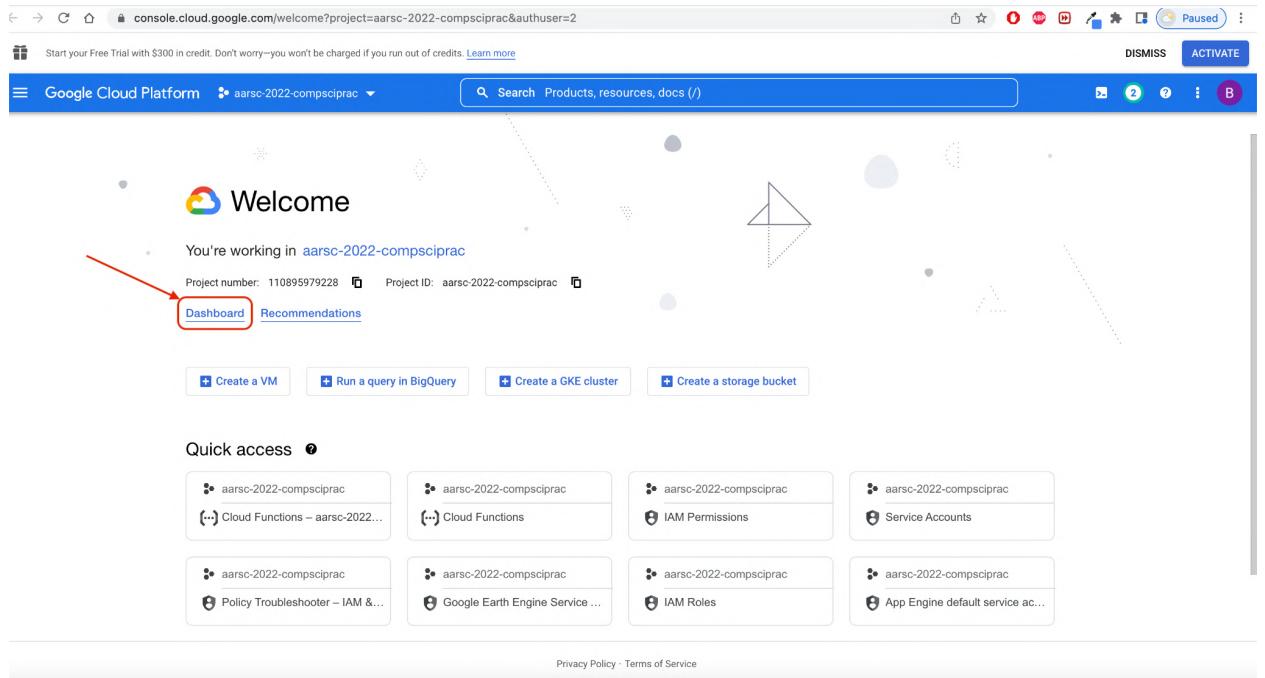
<https://console.cloud.google.com/welcome?project=aarsc-2022-compsciprac>

If you are not logged into an account with access permission, you might see the below screen that has a “select project” drop down at the left of the search bar”:



**Tip :** If you see something similar to the above page, please log into an account with access to the project.

- When logged into an account with access to the project, you will see the project name loaded within the different services; please navigate to the dashboard and click on the link:



- While on the dashboard, head over to the “Cloud Functions” service under the “Resources” section, and click on it:

The screenshot shows the Google Cloud Platform Dashboard. On the left, a sidebar lists various services: IAM & Admin, Billing, APIs & Services, Marketplace, Compute Engine, Cloud Storage, VPC network, Kubernetes Engine, BigQuery, SQL, Security, and Cloud Run. The 'Cloud Functions' section under APIs & Services is highlighted with a red box and an arrow pointing to the 'Event-driven serverless functions' link.

- Next; head over to the existing functions and select the “farm-calculator” function:

The screenshot shows the Google Cloud Platform Functions page. The table lists one function: 'farm-calculator' (1st gen, Region: australia-southeast1, Trigger: HTTP, Runtime: Python 3.9, Memory allocated: 256 MB, Executed function: main\_function, Last deployed: May 30, 2022, 8:06:05 PM). The 'farm-calculator' link is highlighted with a red box.

Environment	Name	Region	Trigger	Runtime	Memory allocated	Executed function	Last deployed	Authentication	Actions
1st gen	farm-calculator	australia-southeast1	HTTP	Python 3.9	256 MB	main_function	May 30, 2022, 8:06:05 PM		⋮

- Once clicked and after the new page loads up, navigate to and click on the “Source” tab, it is two tabs to the right of the “Metrics” tab; On the left side of the screen, we can see 4 files, namely: “main.py”, “requirements.txt”, “farm\_details.json” , and “sa-private-key.json.

farm-calculator - Cloud Function

console.cloud.google.com/functions/details/australia-southeast1/farm-calculator?env=gen1&project=aarsc-2022-compasscrac&tab=source

Cloud Functions Function details EDIT DELETE COPY

farm-calculator Version 12, deployed at May 20, 2022, 9:59:15 ...

METRICS DETAILS SOURCE VARIABLES TRIGGER PERMISSIONS LOGS TESTING

Runtime: Python 3.9 Entry point: main\_.function DOWNLOAD ZIP

main.py

```
1 import ee
2 from datetime import date, timedelta, datetime
3 import time
4 from google.cloud import storage
5 from google.oauth2 import service_account
6 import json
7
8 # Function to calculate NDVI and other vegetation indices for each image in a Sentinel-2 collection
9 def calculate_vegetation_indices(image):
10     # B1 = BLUE
11     # B3 = GREEN
12     # B4 = RED
13     # B5 = Visible and Near Infrared
14     # B8 = Near Infrared
15     # B11 = Short Wave Infrared
16     image = ee.Image(image).copyProperties(image).copyProperties(image,['system:time_start'])
17     image = image.addBands(image.normalizedDifference(['nir','r']).rename('ndvi'))
18     image = image.addBands(image.normalizedDifference(['nir','g']).rename('gsndvi'))
19     image = image.addBands(image.normalizedDifference(['g','r']).rename('grndvi'))
20     image = image.addBands(image.select('nir').divide(image.select('g')).subtract(1).rename('cig'))
21     image = image.addBands(image.expression(
22         '2.5 * (image.select("nir") - 7.5 * b + 1)', {
23             'b1': image.select('nir'),
24             'r': image.select('r'),
25             'g': image.select('g')
26         }).rename('cig'))
27     image = image.addBands(image.select('nir').gt(0).unmask(0).rename('unmasked'))
28     image = image.addBands(image.normalizeDifference(['g','sdr1']).rename('ndvi1'))
29     image = image.addBands(image.normalizeDifference(['nir','sdr1']).rename('lswd1'))
30     image = image.addBands(image.select('nir').divide(image.select('r')).subtract(1).rename('c1r'))
31     return image
32
33
34 # Function to mask clouds using the Sentinel-2 QA band.
35 def mask2Clouds(image):
36     # The Q400 bitmap band contains information on the cloud cover for each pixel.
37     # So select this band
38     qa = image.select('Q400')
```

- **Main.py** – This file contains the python code separated into various functions.

The below code highlighted in red from line 1 to line 6, are the package and library imports utilised by the main function python script of the project;

While, the below *code highlighted in purple* from line 9 to line 31, is the function used to calculate the vegetation indices of interest and assign a corresponding color to the bands of data.

These values and assignments can be changed to accommodate other vegetation indices, and re-assignment of corresponding colors for the bands of data to be collected.

```

1  import ee
2  from datetime import date, timedelta, datetime
3  import time
4  from google.cloud import storage
5  from google.oauth2 import service_account
6  import json
7
8  # Function to calculate NDVI and other vegetation indices for each image in a Sentinel-2 collection
9  def calculate_vegetation_indices(image):
10     # B2 = BLUE
11     # B3 = GREEN
12     # B4 = RED
13     # B5 = Visible and Near Infrared
14     # B8 = Near Infrared
15     # B11 = Short Wave Infrared
16     image = ee.Image(image.divide(10000).copyProperties(image).copyProperties(image, ['system:time_start']))
17     image = image.addBands(image.normalizedDifference(['nir', 'r']).rename('ndvi'))
18     image = image.addBands(image.normalizedDifference(['nir', 'g']).rename('gndvi'))
19     image = image.addBands(image.normalizedDifference(['g', 'r']).rename('grvi'))
20     image = image.addBands(image.select('nir').divide(image.select('g')).subtract(1).rename('cig'))
21     image = image.addBands(image.expression(
22         '2.5 * ((nir-r) / (nir + 6*r - 7.5*b + 1))', {
23             'nir': image.select('nir'),
24             'r': image.select('r'),
25             'b': image.select('b')
26         }).rename('evi'))
27     image = image.addBands(image.select('nir').gt(0).unmask(0).rename('unmasked'))
28     image = image.addBands(image.normalizedDifference(['g', 'swir1']).rename('mdwi'))
29     image = image.addBands(image.normalizedDifference(['nir', 'swir1']).rename('lswi'))
30     image = image.addBands(image.select('nir').divide(image.select('re')).subtract(1).rename('cire'))
31     return image
32
33

```

- Below the calculation of vegetation indices, is the below code from line 35 to line 61 the ‘maskS2clouds’ and ‘mask\_clouds’ functions contained in the orange rectangle are used to mask clouds using the Sentinel-2 QA band and the Sentinel-2 collections; while, the code from line 64 to line 70 contained in the blue rectangle, is a function called “get\_sentinel2\_image\_collection()” that takes in a start\_date, end\_date and an image collection, which it uses to select and rename relevant bands of interest to easily readable values. The mask values and collections can be tweaked, and the relevant bands selected can be changed and renamed as per the interests of the user.

```

34  # Function to mask clouds using the Sentinel-2 QA band.
35  def maskS2clouds(image):
36      # The QA60 bitmask band contains information on the cloud cover for each pixel.
37      # So select this band
38      qa = image.select('QA60')
39
40      # Bits 10 and 11 are clouds and cirrus, respectively.
41      # Use zero fill left shift operators
42      cloudBitMask = 1 << 10
43      cirrusBitMask = 1 << 11
44
45      # Both flags should be set to zero, indicating clear conditions.
46      mask = qa.bitwiseAnd(cloudBitMask).eq(0) and qa.bitwiseAnd(cirrusBitMask).eq(0)
47
48      # Return the masked and scaled data, without the QA bands.
49      # The value returned is either null if clouds are obscuring the
50      # pixel or the actual value if not
51      # return image.updateMask(mask).divide(10000).select("B.*").copyProperties(image, ["system:time_start"])
52      return image.updateMask(mask)
53
54
55  def mask_clouds(img):
56      # masks clouds in Sentinel-2 collections
57      img=ee.Image(img)
58      clouds = ee.Image(img.get('cloud_mask')).select('probability')
59      isNotCloud = clouds.lt(40)
60      return img.updateMask(isNotCloud)
61
62
63  # select the relevant bands and rename to more human-readable heading values
64  def get_sentinel2_image_collection(start_date,end_date, coll='COPERNICUS/S2_HARMONIZED'):
65      sentinel2_image_collection = ee.ImageCollection(coll) \
66          .filterDate(start_date,end_date) \
67          .select(['B2','B3','B4','B5','B8','B6','B7','B8A','B11','B12','QA60'], ['b','g','r','re','nir','re74','re78','re86','swir1','swir2','QA60'])
68
69      sentinel2_cloud_probability = ee.ImageCollection('COPERNICUS/S2_CLOUD_PROBABILITY')\
70          .filterDate(start_date,end_date)
71

```

- Below, the Mask cloud, collection selection-and-renaming functions, are the below functions:

The “get\_timeseries\_feature\_collection()” function is highlighted within the yellow rectangle in the below image and spans code from line 81 to line 97; It is used to get the time series over the feature collection and then return the image collection sample images. It takes in a ‘feature collection’, image collection, jsonfile (geojson) image properties, and scale as arguments.

The function below the “get\_timeseries\_feature\_collection()” function is the “add\_date\_to\_image()” function highlighted in the green rectangle in the image below. It takes in an image as an argument and allows the user to choose a date format and add the corresponding dates to the images.

Both of these functions allow changes to be made to the arguments passed in, the scale of the images, and the format of the date that is appended to the images

```

71     sentinel2_image_collection = ee.Join.saveFirst('cloud_mask').apply(**{
72         'primary': sentinel2_image_collection,
73         'secondary': sentinel2_cloud_probability,
74         'condition': ee.Filter.equals(leftField='system:index', rightField='system:index')
75     })
76     return ee.ImageCollection(sentinel2_image_collection).map(mask2clouds).map(calculate_vegetation_indices)
77
78
79
80     #get time series over a feature collection, general, doesn't filter ic
81     def get_timeseries_feature_collection(feature_collection,image_collection,json_file,image_props=[],scale=10):
82         def get_time_series_single_feature(feature):
83             geom = feature.geometry();
84             def sample_image(image):
85                 return feature.set(image.reduceRegion(**{'reducer':ee.Reducer.mean(), 'geometry':geom, 'scale':scale})) \
86                     .setGeometry(geom.centroid(10)) \
87                     .set(image.toDictionary(image_props)) \
88                     .set('area_gee',geom.area(1).divide(10000)) \
89                     .set('geojson_filename',json_file)
90             samples = image_collection.map(sample_image)
91             return samples
92
93         #get time series for a single feature
94         time_series = feature_collection.map(get_time_series_single_feature)
95
96         #will be a collection of collections, so need to flatten to a collection
97         return time_series.flatten()
98
99
100    def add_date_to_image(image):
101        tz = 'Australia/Sydney'
102        date = ee.Date(image.get('system:time_start'))
103        au_formatted_date = date.format(**{'format':'yyyy-MM-dd','timeZone':tz})
104        return image.set({
105            'date':au_formatted_date,
106        });
107

```

The “main\_function” function handles the initialisation of the service account and private key, it also contains the setting for the timeframe or time period which CropWatch-RDSIF will collect data within. It includes the loading of the geojson file to obtain the farm details from, and finally; a function to add a description, set an output destination (databucket), and define the file format of the export (in this case ‘.csv’).

*Line 109 to line 116*, is where the service account and private key for the project is initialised. - This can be edited to initialise another project and another private key if needed.

```

109    def main_function(request):
110        #initialize earth engine using json key for service account
111        service_account = 'google-earth-engine-service-ac@aarsc-2022-compsciprac.iam.gserviceaccount.com'
112        privateKey = 'sa-private-key.json'
113
114        # authenticate using service account
115        credentials = ee.ServiceAccountCredentials(service_account, privateKey)
116        ee.Initialize(credentials)
117

```

*Line 119 -120* includes a variable called “start\_date”, which sets the start of the timeframe for the Google Earth images data. The variable “end\_date” determines the most current date you want the function to retrieve data up to:

```
116  # set the start and end dates for the code
117  # this should be moved to arguments in the future
118  # start with a 2 year timeframe (730 days)
119  start_date = (date.today() - timedelta(days=730)).strftime("%Y-%m-%d")
120  end_date = date.today().strftime("%Y-%m-%d")
121
```

“start\_date” is currently set in this screenshot to today minus 730 days (2 years) but can be adjusted to any other value. The currently deployed, default adjusted value was set to 1095 days (3 years) as can be seen in the screenshot below:

```
128  # set the start and end dates for the code
129  # this should be moved to arguments in the future
130  # start with a 3 year timeframe (1095 days)
131  start_date = (date.today() - timedelta(days=1095)).strftime("%Y-%m-%d")
132  end_date = date.today().strftime("%Y-%m-%d")
```

Hence; 3 years of data from the current date can be obtained and exported by CropWatch-RDSIF’s farm-calculator function.

The code on *line 118 to line 126*, provides the option to get an optional POST argument for the json file name:

```
118  # get optional POST argument for the json filename
119  request_json = request.get_json(silent=True)
120  request_args = request.args
121  if request_json and 'json_filename' in request_json:
122      json_filename = request_json['json_filename']
123  elif request_args and 'json_filename' in request_args:
124      json_filename = request_args['json_filename']
125  else:
126      json_filename = 'farm_details.json'
```

The code on *line 134 to line 138*, allows the loading of the geojson file that is passed into the previous argument.

```
134  # load a geojson file to get farm details
135  # json_filename = 'farm_details.json'
136  json_file = open(json_filename)
137  feature_collection_json = json.load(json_file)
138  json_file.close()
```

*The code on line 140 to line 151*, handles the creation of the feature collection based on the geojson data supplied from the geojson file. The farm boundaries are then extracted from this feature collection and used in combination with Sentinel2 satellite images within the earlier supplied date range (start\_date , end\_date) to obtain the image statistics at each time for each element in the feature collection.

```

148 # create the feature collection based on the geojson supplied
149 feature_collection = ee.FeatureCollection(feature_collection_json)
150
151 # extract the farm boundaries from the feature collection
152 geom = feature_collection.geometry().bounds()
153
154 # create an image collection from the sentinel2 satellite images using the date ranges and geo boundaries
155 # - collection of S2 images covering the feature collection of interest
156 sentinel2_image_collection = get_sentinel2_image_collection(start_date,end_date).filterBounds(geom).map(add_date_to_image)
157
158 #gets image statistics at each time over each field defined in feature collection
159 time_series = get_timeseries_feature_collection(feature_collection,sentinel2_image_collection,json_filename,image_props=['date','ndvi','gndvi','grvi','cig','mndwi','lswi','cire'],scale=10)
160
161
162
163
164

```

*The code on line 156* sets the description field for the data export. The system utilises this field to set the name of the exported CSV file. It is currently set to farm\_data\_export\_[current date/time]; however, this can be changed to any other value.

The code from line 154 to line 163 is utilised to add a description field for the data being exported, set an output destination (databucket), and define the file format of the export (in this case ‘.csv’) as can be observed below:

```

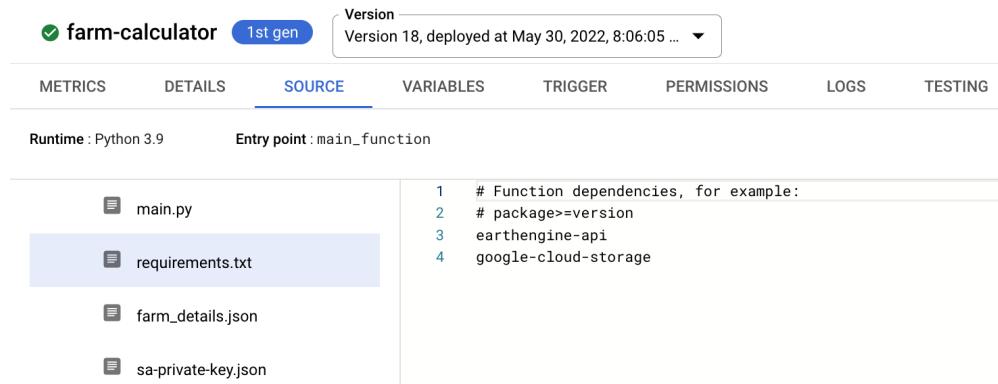
153 # Function to export statistical values per ID with the information of day, month and year
154 task = ee.batch.Export.table.toCloudStorage(
155   collection=time_series,
156   description='farm_data_export_'+str(datetime.now().year) + "-" + str(datetime.now().month) + "-" + str(datetime.now().day) + " " + str(datetime.now().hour) + ":" + str(datetime.now().minute),
157   outputBucket='aarsc-2022-compsciprac-csv-databucket',
158   fileFormat='csv'
159   #selectors=['date','id','ndvi','gndvi','grvi','cig','mndwi','lswi','cire']
160 )
161 task.start()
162
163 return "completed"
164

```

- Once the file is exported, the filename is set and exported to the databucket as: ‘farm\_data\_export\_[date and time of the export]’ in a ‘.csv’ format.

## The other documents within the farm-calculator are as follows:

- **Requirements.txt** – This file sets out the relevant dependencies for the Python code. Currently, the code relies on the Google Earth Engine API and Google Cloud Storage



farm-calculator 1st gen Version Version 18, deployed at May 30, 2022, 8:06:05 ...

METRICS DETAILS SOURCE VARIABLES TRIGGER PERMISSIONS LOGS TESTING

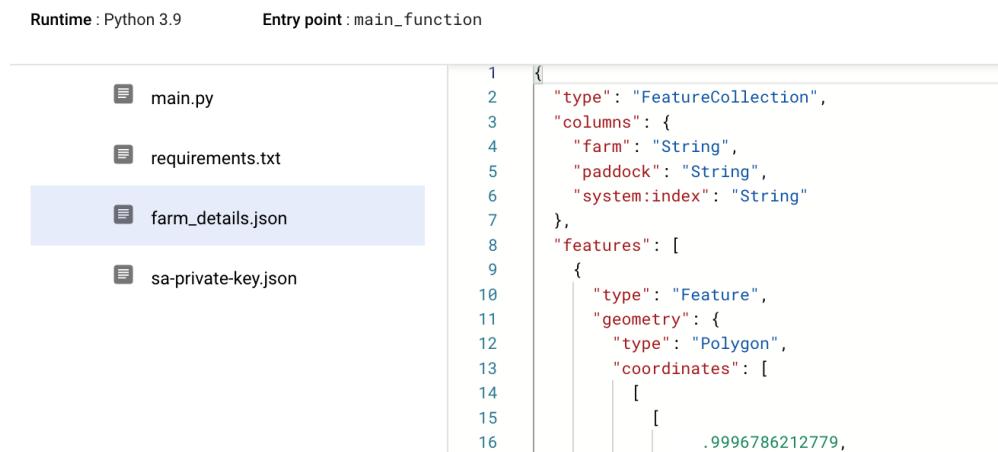
Runtime : Python 3.9 Entry point : main\_function

main.py requirements.txt farm\_details.json sa-private-key.json

```
1 # Function dependencies, for example:  
2 # package>=version  
3 earthengine-api  
4 google-cloud-storage
```

- **Farm\_details.json** – This file sets out the boundaries of the farms to be examined with the code. This file contains GeoJSON data saved in JSON format.

### Coordinates redacted example:



Runtime : Python 3.9 Entry point : main\_function

main.py requirements.txt farm\_details.json sa-private-key.json

```
1 {  
2   "type": "FeatureCollection",  
3   "columns": {  
4     "farm": "String",  
5     "paddock": "String",  
6     "system:index": "String"  
7   },  
8   "features": [  
9     {  
10       "type": "Feature",  
11       "geometry": {  
12         "type": "Polygon",  
13         "coordinates": [  
14           [  
15             [  
16               .9996786212779,
```

The screenshot shows the Google Cloud Functions console for the 'farm-calculator' function. The 'SOURCE' tab is active, showing the code structure and a preview of the 'farm\_details.json' file content. The code is a Python script with line numbers 38 to 55. The 'farm\_details.json' file is highlighted, showing its JSON content which includes a feature collection with a single polygon feature and its properties.

```

38     ]
39     ]
40   },
41   "id": "0",
42   "properties": {
43     "farm": "f1",
44     "paddock": "p1"
45   }
46 },
47 {
48   "type": "Feature",
49   "geometry": {
50     "type": "Polygon",
51     "coordinates": [
52       [
53         [
54           [
55             .99624539373883,

```

To examine a different set of farms, this file can be either replaced with another valid FeatureCollection in JSON format or adjustments to the python code on line 123, within the main function can be made.

- **Sa-private-key** – This file provides the private key for the service account used to run the Python code. If a different service account is used, this file will need to be updated.

**Tip :** This “Sa-private-key” file, is a confidential file of which its contents cannot be shown to non-members of the team and should not be uploaded to the open source github repository.

**There are a number of ways to run the cloud function:**

1. Use the HTTP trigger URL  
(<https://australia-southeast1-aarsc-2022-compsciprac.cloudfunctions.net/farm-calculator>).  
Permissions are required to run the function this way; otherwise, access will be denied.
2. Open the function, select the ‘Testing’ tab and then click the ‘Test the Function’ button .

Google Cloud Platform > aarsc-2022-comsciprac > Cloud Functions > farm-calculator (1st gen) > Function details > EDIT > DELETE > COPY

farm-calculator Version Version 12, deployed at May 20, 2022, 9:59:15...

METRICS DETAILS SOURCE VARIABLES TRIGGER PERMISSIONS LOGS TESTING

Configure Triggering Event ⓘ

Press Alt+F1 for Accessibility Options.

1

TEST THE FUNCTION

Testing in the Cloud Console has a 5 minute timeout. Note that this is different from the limit set in the function configuration.

Test Command TEST IN CLOUD SHELL

```
curl -m 70 -X POST https://australia-southeast1-aarsc-2022-comsciprac.cloudfunctions.net/farm-calculator \
-H "Authorization: bearer $(gcloud auth print-identity-token)" \
-H "Content-Type:application/json" \
-d "{}"
```

- Below is what shows up when the task is completed

Google Cloud Platform > aarsc-2022-comsciprac > Cloud Functions > farm-calculator (1st gen) > Function details > EDIT > DELETE > COPY

farm-calculator Version Version 12, deployed at May 20, 2022, 9:59:15...

METRICS DETAILS SOURCE VARIABLES TRIGGER PERMISSIONS LOGS TESTING

Configure Triggering Event ⓘ

Press Alt+F1 for Accessibility Options.

1

TEST THE FUNCTION

Testing in the Cloud Console has a 5 minute timeout. Note that this is different from the limit set in the function configuration.

Output Complete

S completed

Logs Fetched (up to 100 entries). View all logs

Scanned up to 3/7/22, 10:59 AM. Scanned 563 B.

2022-05-24T05:26:53.384981787Z farm-calculator tc2ah36e49ek Function execution started

- Next, navigate to the cloud storage (databucket) by clicking the menu icon at the top left of the page, next to “Google Cloud Platform” header, then scroll down to the “Storage” section and click on “Cloud Storage”, then “browse”:

Google Cloud Platform sidebar (left):

- Security
- Anthos
- COMPUTE
  - Compute Engine
  - Kubernetes Engine
  - VMware Engine
  - Distributed Cloud
- SERVERLESS
  - Cloud Run
  - Cloud Functions
  - App Engine
- STORAGE
  - Filestore
  - Cloud Storage
  - Data Transfer
- DATABASES

Context menu (right):

- Browser
- Monitoring
- Settings

- This will load the cloud storage page which will have the databucket that is being used within the project at the top of the list (only selected below for reference purposes), click on it:

Google Cloud Platform Cloud Storage Browser page:

Cloud Storage sidebar (left):

- Browser
- Monitoring
- Settings

Cloud Storage table (right):

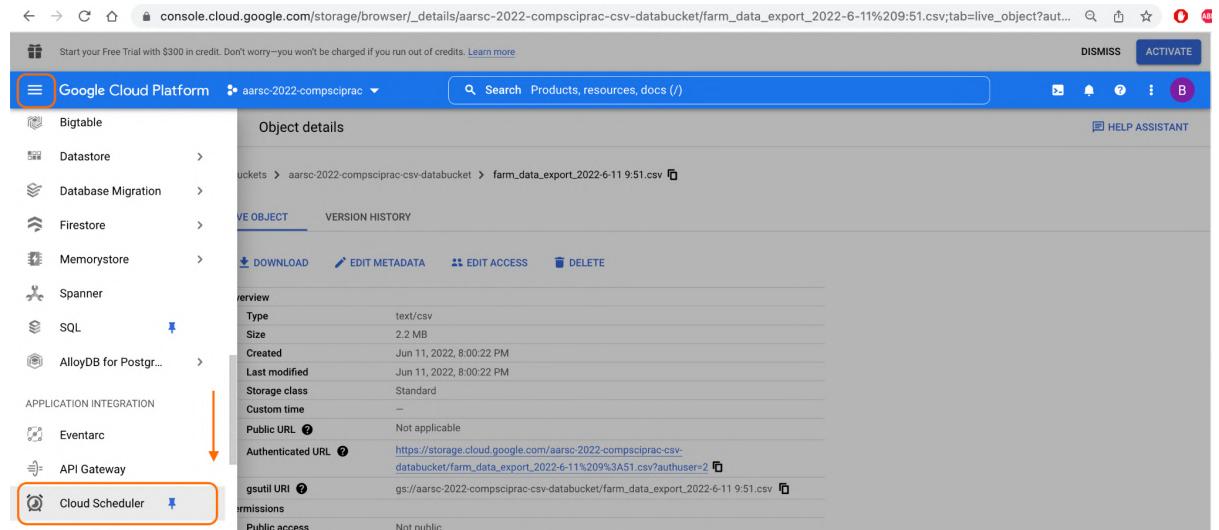
Name	Created	Location type	Location	Default storage class	Last modified	Public access	Access control	Protection
<input checked="" type="checkbox"/> aarsc-2022-compsciprac-csv-database	Apr 30, 2022, 4:21:54 PM	Region	australia-southeast1	Standard	Apr 30, 2022, 4:21:54 PM	Not public	Uniform	None
<input type="checkbox"/> asia.artifacts.aarsc-2022-compsciprac	Apr 6, 2022, 8:32:40 PM	Multi-region	asia	Standard	Apr 6, 2022, 8:32:40 PM	Subject to object ACLs	Fine-grained	None
<input type="checkbox"/> gcf-sources-110895979228-australia...	Apr 6, 2022, 8:31:55 PM	Region	australia-southeast1	Standard	Apr 6, 2022, 8:31:55 PM	Not public	Uniform	None
<input type="checkbox"/> gcf-sources-110895979228-us-central1	Mar 23, 2022, 8:01:42 AM	Region	us-central1	Standard	Mar 23, 2022, 8:01:42 AM	Not public	Uniform	None
<input type="checkbox"/> loc000	Mar 15, 2022, 8:55:51 AM	Region	australia-southeast1	Standard	Mar 15, 2022, 8:55:51 AM	Not public	Uniform	None
<input type="checkbox"/> us.artifacts.aarsc-2022-compsciprac...	Mar 23, 2022, 8:02:11 AM	Multi-region	us	Standard	Mar 23, 2022, 8:02:11 AM	Subject to object ACLs	Fine-grained	None

- The page that loads has the most current exported data (only selected below for reference purposes) (based on service usage, the time it takes to appear in the cloud storage might vary, based on if you are exporting at peak hours and or if your internet connection is not as fast):

Clicking on the file, shows further information about the file as can be seen below, and provides the option to edit access, change metadata, delete the file or to download the file:

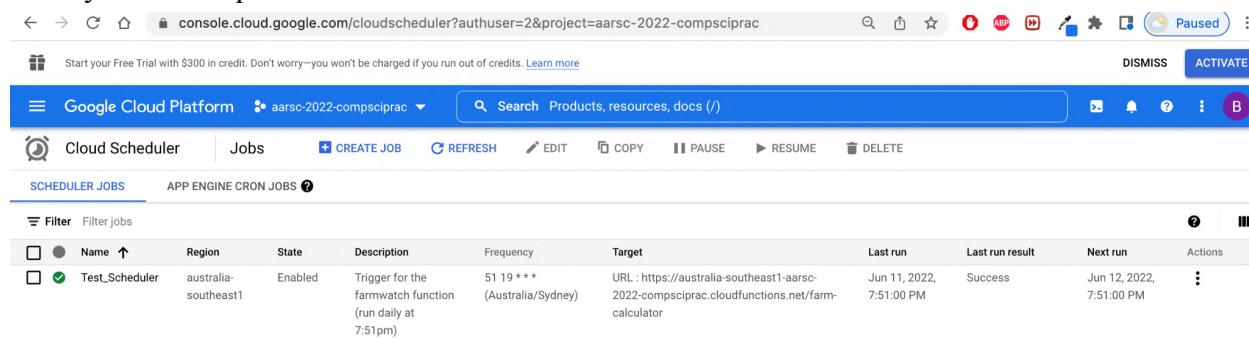
3. The files are processed using Google Cloud Scheduler jobs. A scheduled task has been set up to run the function daily; however, the scheduler will also allow a manual trigger of the task by clicking the ‘Run Now’ button.

- Click on the menu icon at the top left of the page, next to “Google Cloud Platform” header, then scroll down to the “Application Integration” section and click on “Cloud Scheduler”:



The screenshot shows the Google Cloud Platform Storage browser. The sidebar on the left lists various services: Bigtable, Datastore, Database Migration, Firestore, Memorystore, Spanner, SQL, and AlloyDB for PostgreSQL. Below these, under 'APPLICATION INTEGRATION', are Eventarc and API Gateway. At the bottom of this list is 'Cloud Scheduler', which is highlighted with a red box and an arrow pointing to the main content area. The main content area shows 'Object details' for a file named 'farm\_data\_export\_2022-6-11 9:51.csv'. The 'OVERVIEW' section includes details like Type: text/csv, Size: 2.2 MB, Created: Jun 11, 2022, 8:00:22 PM, and Last modified: Jun 11, 2022, 8:00:22 PM. It also shows Public URL and Authenticated URL.

- Next, the page loads up and shows the aforementioned existing schedule that has already been set up:



The screenshot shows the Google Cloud Platform Cloud Scheduler jobs list. The top navigation bar includes 'Google Cloud Platform', 'aarsc-2022-compsciprac', a search bar, and various icons. The main area has tabs for 'SCHEDULER JOBS' and 'APP ENGINE CRON JOBS'. Below is a table with columns: Name, Region, State, Description, Frequency, Target, Last run, Last run result, Next run, and Actions. One job is listed: 'Test\_Scheduler' (Region: australia-southeast1, State: Enabled, Description: Trigger for the farmwatch function (Australia/Sydney), Frequency: 51 19 \* \* \*, Target: URL: https://australia-southeast1-aarsc-2022-compsciprac.cloudfunctions.net/farm-calculator). The 'Last run' is Jun 11, 2022, 7:51:00 PM, and the 'Last run result' is Success. The 'Next run' is Jun 12, 2022, 7:51:00 PM. The 'Actions' column shows a three-dot menu icon.

- Changes can be made to this schedule by clicking on it and making adjustments to the parameters below, please observe that the cloud scheduler in this “test\_schedule” is currently set to trigger and run the CropWatch farm-calculator function every day at 7:51 pm, AEST (Australian Eastern Standard Time):

The screenshot shows the Google Cloud Platform Cloud Scheduler interface. At the top, the navigation bar includes the Google Cloud Platform logo, the project name 'aarsc-2022-compsciprac', and a search icon. Below the navigation bar, the 'Cloud Scheduler' logo and the name 'Test\_Scheduler' are displayed, with a back arrow icon to its left. The main content area is titled 'Define the schedule' and contains the following fields:

- Region:** australia-southeast1
- Description:** Trigger for the farmwatch function (run daily at 7:51pm)
- Frequency \***: 51 19 \* \* \*

Schedules are specified using unix-cron format. E.g. every minute: "\* \* \* \* \*", every 3 hours: "0 \*/3 \* \* \*", every monday at 9:00: "0 9 \* \* 1". [Learn more](#)

- Timezone \***: Australian Eastern Standard Time (AEST)

Below the configuration fields, there is a 'CONTINUE' button. At the bottom of the page, there are 'UPDATE' and 'CANCEL' buttons.

- The current execution configuration of the cloud scheduler is displayed below, and can be adjusted as pleased:

Google Cloud Platform aarsc-2022-compscicprac Test\_Scheduler

Configure the execution

Target type \* HTTP

URL \* https://australia-southeast1-aarsc-2022-compscicprac.cloudfunctions.net/farm-

HTTP method POST

HTTP headers

Some headers are set to default values or removed by Cloud Scheduler. [Learn more](#)

Name 1 \* User-Agent Value 1 Google-Cloud-Scheduler

+ ADD A HEADER

Body

Auth header Add OIDC token

Service account \* Google Earth Engine Service Account SD

This service account must have permission to invoke the target. For example, the Cloud Functions Invoker role is required to schedule a Cloud Function. [Learn more](#)

Audience https://australia-southeast1-aarsc-2022-compscicprac.cloudfunctions.net/farm-

Audience limits recipients for the OIDC token. Typically, the job's target URL (without any url parameters). If not specified, by default, Cloud Scheduler will use the whole URL, including request parameters, as the Audience.

CONTINUE

- The optional settings that handle the retrying of the scheduled function in the event of a job not completing successfully are shown below; and can be adjusted as pleased:

Google Cloud Platform aarsc-2022-compscicrac

Cloud Scheduler Test\_Scheduler

Audience limits recipients for the OIDC token. Typically, the job's target URL (without any url parameters). If not specified, by default, Cloud Scheduler will use the whole URL, including request parameters, as the Audience.

CONTINUE

Configure optional settings

**Retry config**

If a job does not complete successfully, it is retried, with exponential backoff, according to the settings in retry config. [Learn more](#)

Max retry attempts	0
Maximum number of retry attempts for a failed job	
Max retry duration *	0s
Time limit for retrying a failed job, 0s means unlimited	
Min backoff duration *	5s
Max backoff duration *	1h
Minimum time to wait before retrying a job after it fails	
Max doublings *	5
The time between retries will double max doublings times	

**Attempt deadline config**

Attempt deadline

3m

Leave empty to reset duration to the default 3 minutes.

UPDATE CANCEL

- To create a new job; click on the “Cloud Scheduler” tab at the top left of the page, then click on the “Create Job” tab located at the top, near the middle of the page:

Google Cloud Platform aarsc-2022-compscicrac

Cloud Scheduler 1. Jobs 2. CREATE JOB

REFRESH EDIT COPY PAUSE RESUME DELETE

SCHEDULER JOBS APP ENGINE CRON JOBS

Filter Filter jobs

Name	Region	State	Description	Frequency	Target	Last run	Last run result	Next run	Actions
Test_Scheduler	australia-southeast1	Enabled	Trigger for the farmwatch function (run daily at 7:51pm)	51 19 * * *	URL : https://australia-southeast1-aarsc-2022-compscicrac.cloudfunctions.net/farm-calculator	Jun 11, 2022, 7:51:00 PM	Success	Jun 12, 2022, 7:51:00 PM	⋮

- Next you will be provided an empty form to fill in the parameters with required information to create the scheduled job (**Please note that:** an example of the required information needed, has already been shown in the above steps within the above existing scheduled job named “Test\_Scheduler”):

Google Cloud Platform aarsc-2022-compsciprac Create a job

Cloud Scheduler Create a job

- Define the schedule

Name \*

! Name is required

Region \*

! Region is required

Description

Frequency \*

! Frequency is required

Timezone \*

! Timezone is required

CONTINUE

- Configure the execution
- Configure optional settings

CREATE CANCEL

This concludes the options and processes directly related with the cloud-function.

## **CropWatch-RDSIF: Exporting farm data to the Database (BigQuery)**

After data gets exported from the CropWatch-RDSIF farm-calculator’s Main.py script’s main function into the “Aarsc-2022-compsciprac-csv-databucket”, the project’s Google Cloud Storage Bucket; the below processes commence:

### **Daily Data Transfer from the Databucket**

A daily Google Data Transfer called “Daily Load 2” runs at 1:00 AM AEST and transfers all files that meet the following criteria

gs://aarsc-2022-compsciprac-csv-databucket/farm\_data\_export\_\*.csv to table

`'aarsc-2022-compsciprac.aarsc_import_data.Daily_Import2'`

All files that meet the file name criteria are appended to the Daily\_Import2 table. At this point there are no checks for duplicates carried out on the files.

Details of this transfer can be amended by clicking on the transfer, then Configuration / Edit. In this section you can select for the files not to be deleted after processing. This can be done by clicking the data transfer, clicking on Configuration/Edit and then clicking off the Delete Source Files on Transfer button. Note that if you wish to reprocess the files you will need to rename them prior to reprocessing. The timing of the transfer as well as some of the file configuration details can be altered in this section.

Every day at 1:00 AM Australia/Sydney

## Destination settings

Select the destination for the transfer data

Dataset \*  
aarsc\_import\_data

## Data source details

Destination table  
Daily\_Import2

Cloud Storage URI  
aarsc-2022-compsciprac-csv-databucket/farm\_data\_export\_\*.c BROWSE

Write preference  
APPEND

Delete source files after transfer

File format \*  
CSV

The data transfer can be run on demand by selecting the data transfer and then selecting Run Transfer now at the top right of the screen.

Transfer details		RUN HISTORY	CONFIGURATION	EDIT	DELETE	DISABLE	RUN TRANSFER NOW	MORE
<b>Daily Load 2</b>								
Schedule (UTC) Target date for next run								
every day 15:00	June 5, 2022 at 1:00:00 AM UTC+10							
<input type="button" value="Filter"/> Filter transfer runs								
Run date	Schedule time	Summary						
June 3, 2022	June 4, 2022 at 1:00:00 AM UTC+10	The transfer run has completed successfully.						
June 2, 2022	June 3, 2022 at 1:00:00 AM UTC+10	The transfer run has completed successfully.						
June 1, 2022	June 2, 2022 at 1:00:00 AM UTC+10	The transfer run has completed successfully.						
May 31, 2022	June 1, 2022 at 1:00:00 AM UTC+10	The transfer run has completed successfully.						
May 30, 2022	May 31, 2022 at 1:00:00 AM UTC+10	The transfer run has completed successfully.						
May 29, 2022	May 30, 2022 at 8:06:42 AM UTC+10	The transfer run has completed successfully.						
May 29, 2022	May 30, 2022 at 1:00:00 AM UTC+10	The transfer run has completed successfully.						
May 29, 2022	May 29, 2022 at 7:04:06 PM UTC+10	The transfer run has completed successfully.						
May 29, 2022	May 29, 2022 at 6:57:38 PM UTC+10	The transfer run has completed successfully.						
May 29, 2022	May 29, 2022 at 6:12:11 PM UTC+10	The transfer run has completed successfully.						

## Scheduled Daily SQL

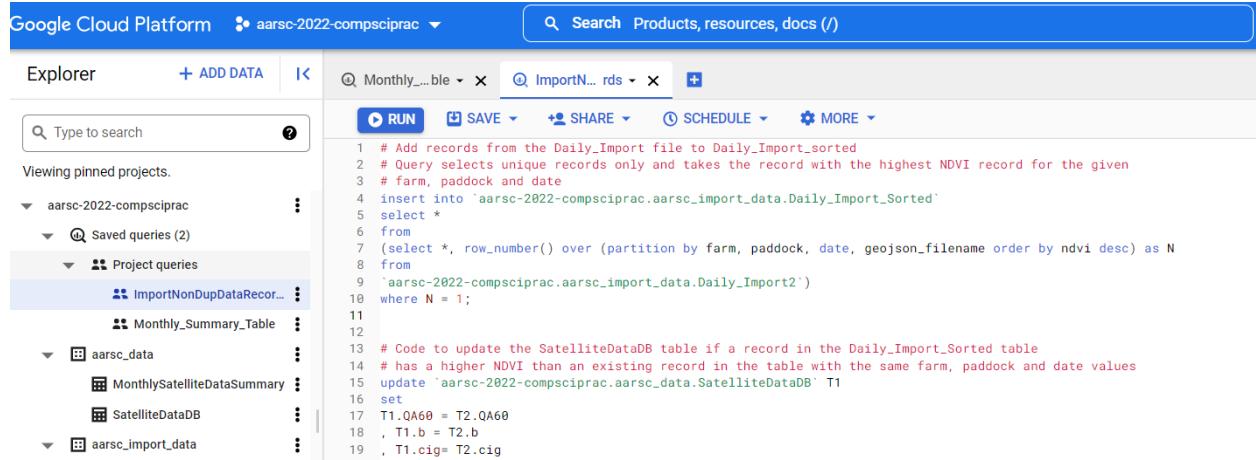
At 2:00 AM each day a scheduled SQL query runs, DailySatelliteDataLoad. It does the following:

- Selects unique records (based on farm, paddock, date and geojson\_filename) in the Daily\_Import file to copy to the Daily\_Import\_sorted file. Where duplicates exist the record with the highest ndvi record is selected.
- For each record in the Daily\_Import\_sorted table a check is done to see if a record with the same farm, paddock, date and geojson\_filename exists in the SatelliteDataDB table. If a record exists and the ndvi is less than the ndvi in the Daily\_Import\_sorted table an update is made to SatelliteDataDB to include the values for the record with the higher ndvi from Daily\_Import\_sorted
- For each record in the Daily\_Import\_sorted table if a record with the same farm, paddock, date and geojson\_filename doesn't exist the record is added to the SatelliteDataDB table
- The content from both
  - `aarasc-2022-compsciprac.aarsc\_import\_data.Daily\_Import`
  - `aarasc-2022-compsciprac.aarsc\_import\_data.Daily\_Import\_sorted`
 is deleted in preparation for the following nightly run

The scheduled query can be run on demand by selecting the query and then clicking on Schedule Backfill in the top right hand corner of the screen.

The screenshot shows the Google Cloud Platform BigQuery interface. At the top, it says 'Google Cloud Platform' and 'aarasc-2022-compsciprac'. There is a search bar with 'Search Products, resources, docs ()'. Below the search bar are buttons for 'EDIT', 'DELETE', 'DISABLE', 'SCHEDULE BACKFILL', and 'MORE'. The main area is titled 'DailySatelliteDataLoad'. It shows a 'Schedule (UTC)' section with 'Target date for next run' set to 'every day 16:00' and 'June 6, 2022 at 2:00:00 AM UTC+10'. Below this is a 'RUN HISTORY' section with a single run entry. The run entry shows 'Run date' as 'June 6, 2022' and 'Schedule time' as 'UTC+10'. The 'SUMMARY' section below says 'No rows to display'.

The underlying SQL can also be run manually by selecting the ImportNonDupDataRecords within the BigQuery database. By highlighting the section of the SQL to be run and clicking on Run at the top the SQL can be run in sections.



The screenshot shows the Google Cloud Platform BigQuery interface. The top navigation bar includes 'Google Cloud Platform', the project name 'aarsc-2022-compsciprac', and a search bar. The main area is divided into two sections: 'Explorer' on the left and the query editor on the right.

**Explorer:** Shows 'Viewing pinned projects' with a list of datasets and tables. The 'aarsc-2022-compsciprac' dataset is expanded, showing 'Saved queries (2)'. One query, 'ImportNonDupDataRecor...', is selected and expanded to show its details.

**Query Editor:** The query 'ImportN... rds' is selected. The SQL code is as follows:

```

1 # Add records from the Daily_Import file to Daily_Import_sorted
2 # Query selects unique records only and takes the record with the highest NDVI record for the given
3 # farm, paddock and date
4 insert into `aarsc-2022-compsciprac.aarsc_import_data.Daily_Import_Sorted`
5 select *
6 from
7 (select *, row_number() over (partition by farm, paddock, date, geojson_filename order by ndvi desc) as N
8 from
9 `aarsc-2022-compsciprac.aarsc_import_data.Daily_Import2`
10 where N = 1;
11
12
13 # Code to update the SatelliteDataDB table if a record in the Daily_Import_Sorted table
14 # has a higher NDVI than an existing record in the table with the same farm, paddock and date values
15 update `aarsc-2022-compsciprac.aarsc_data.SatelliteDataDB` T1
16 set
17 T1.QA60 = T2.QA60
18 , T1.b = T2.b
19 , T1.cig= T2.cig

```

**Please note that:** changes made to the underlying SQL do not get passed through to the schedule query. To update the scheduled query you will need to create a new scheduled query by clicking on Schedule / Create new Schedule. You will also need to ensure that the data location is consistent with where the data is currently being stored, in this case australia-southeast1 (Sydney).

New scheduled query

Destination for query results

Set a destination table for query results

Dataset

Table Id

Destination table partitioning field

Destination table write preference

Append to table

Overwrite table

**Data location**

Filter Type to filter

Region	Location	CO2
A	northamerica-northeast1 (Montréal)	Low CO2
	northamerica-northeast2 (Toronto)	Low CO2
N	southamerica-east1 (São Paulo)	Low CO2
	southamerica-west1 (Santiago)	
	australia-southeast1 (Sydney)	
	australia-southeast2 (Melbourne)	
	aws-us-east-1	
	azure-eastus2	

SAVE CANCEL

## Scheduled Monthly SQL

At 3am on the 4th day of each month a monthly scheduled SQL is run called MonthlySummaryUpdate. This process deletes the content of the MonthlySatelliteDataSummary table and recreates it based on the data available in the SatelliteDataDB table. The summary table contains monthly averages, minimums and maximums, standard deviation and the upper and lower 95% confidence intervals for each of the indices in Satellite Data DB by farm, paddock and geojson\_filename.

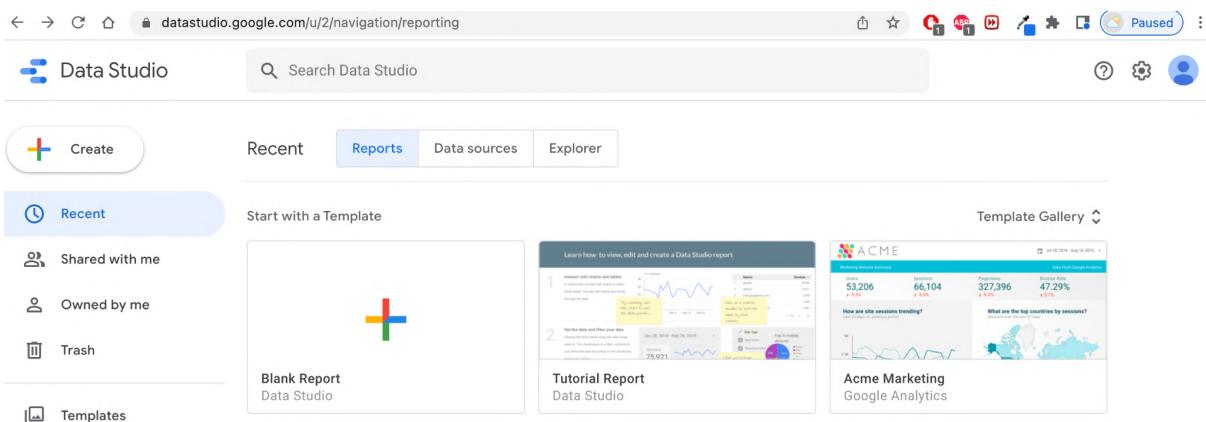
# CropWatch-RDSIF: Visualizing Database information in DataStudio

## Google DataStudio process

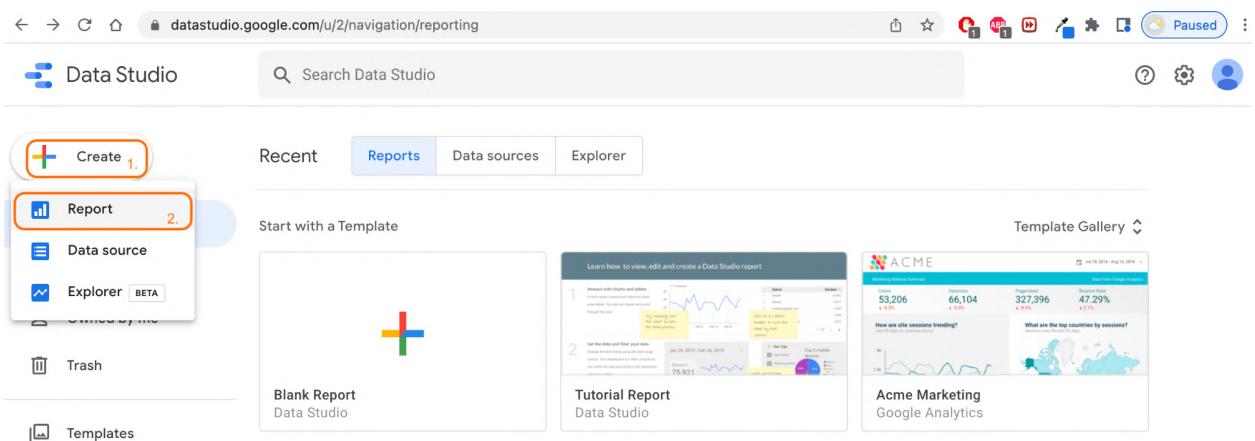
The end product of the CropWatch-RDSIF application process is to create reports that construct visual representations of the data collected and processed, to provide farmers valuable insights on the crop biomass and crop health of the crops in their paddocks/ farms over a given period of time up to 3 years front the current date.

The following steps should be followed to create these visual representations of the data collected and processed:

1. Go to <https://datastudio.google.com/>



2. Click on 'Create', then select 'Report':



3. Add data to report, choose 'connect to data', then click on 'BigQuery'.

Untitled Report

Add data to report

Connect to data My data sources

Google Connectors (22)  
Connectors built and supported by Data Studio [Learn more](#)

Google Analytics By Google Connect to Google Analytics

Google Ads By Google Connect to Google Ads performance report data.

Google Sheets By Google Connect to Google Sheets

BigQuery By Google Connect to BigQuery tables and custom queries.

4. The select project 'aarsc-2022-compsciprac' 'aarsc\_data' 'MonthlySatelliteDataSummary' and 'SatelliteDataDB'

← Add data to report

⚡ Make your BigQuery reports load even faster with BigQuery BI Engine. [Learn More](#)

**BigQuery**  
By Google

BigQuery is Google's fully managed, petabyte scale, low-cost analytics data warehouse. BigQuery charges for querying/processing of data. Those queries are charged to the credit card of the billing project.

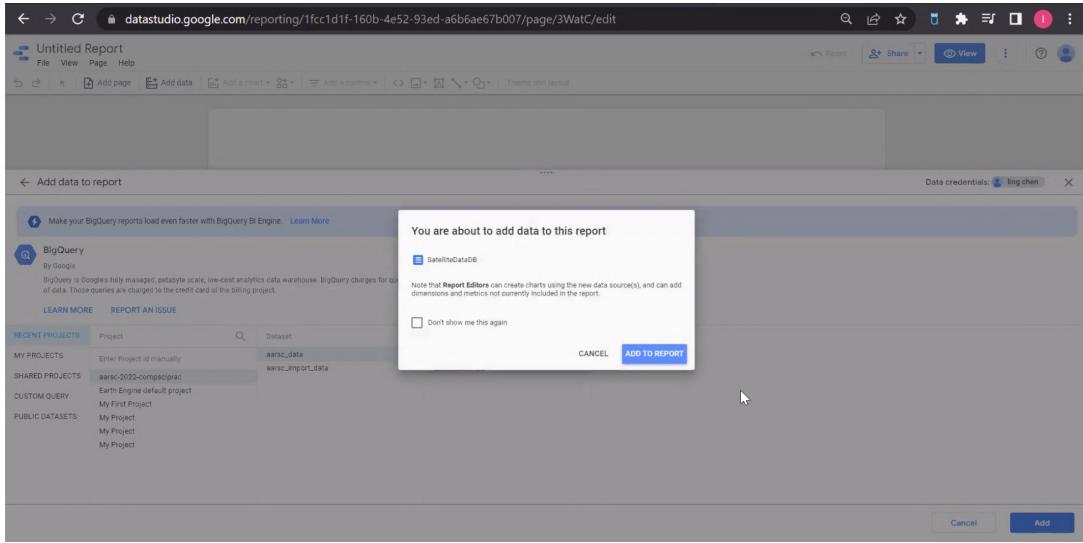
[LEARN MORE](#) [REPORT AN ISSUE](#)

RECENT PROJECTS	Project	Dataset	Table
MY PROJECTS	Enter Project Id manually	aarsc_data	MonthlySatelliteDataSummary
SHARED PROJECTS	aarsc-2022-compsciprac	aarsc_import_data	SatelliteDataDB
CUSTOM QUERY	Earth Engine default project		
PUBLIC DATASETS	My First Project		
	My Project		

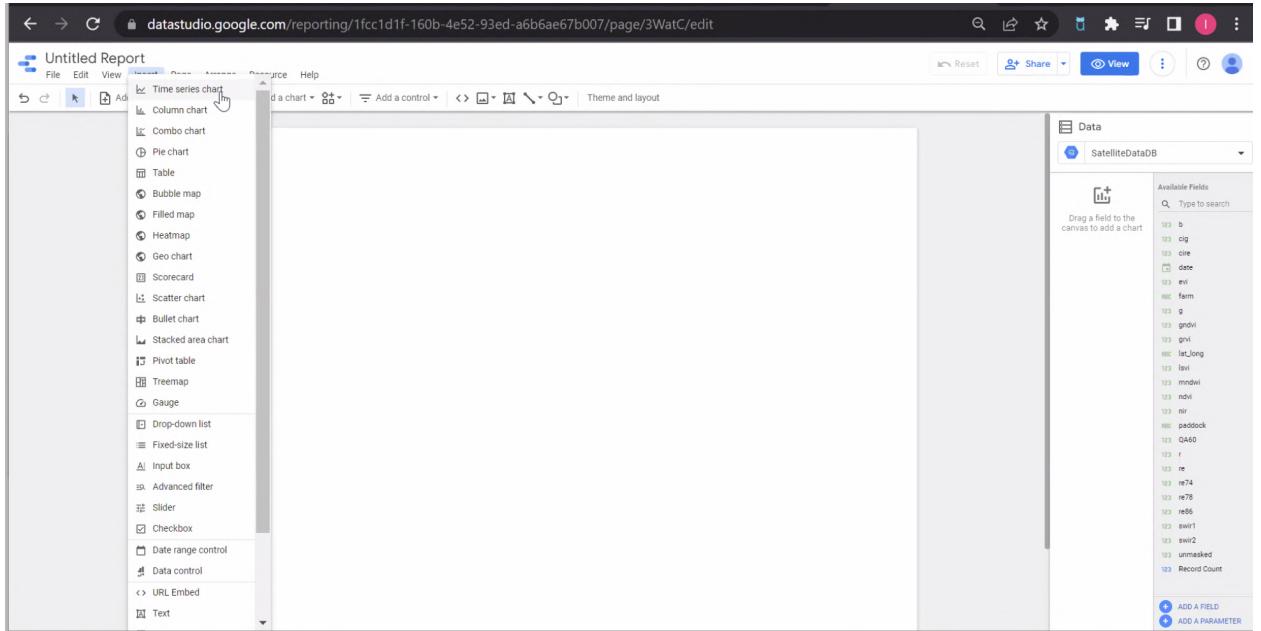
When the report is created, if the below is shown, click on the “Add data” tab at the top:

Select the data to add:

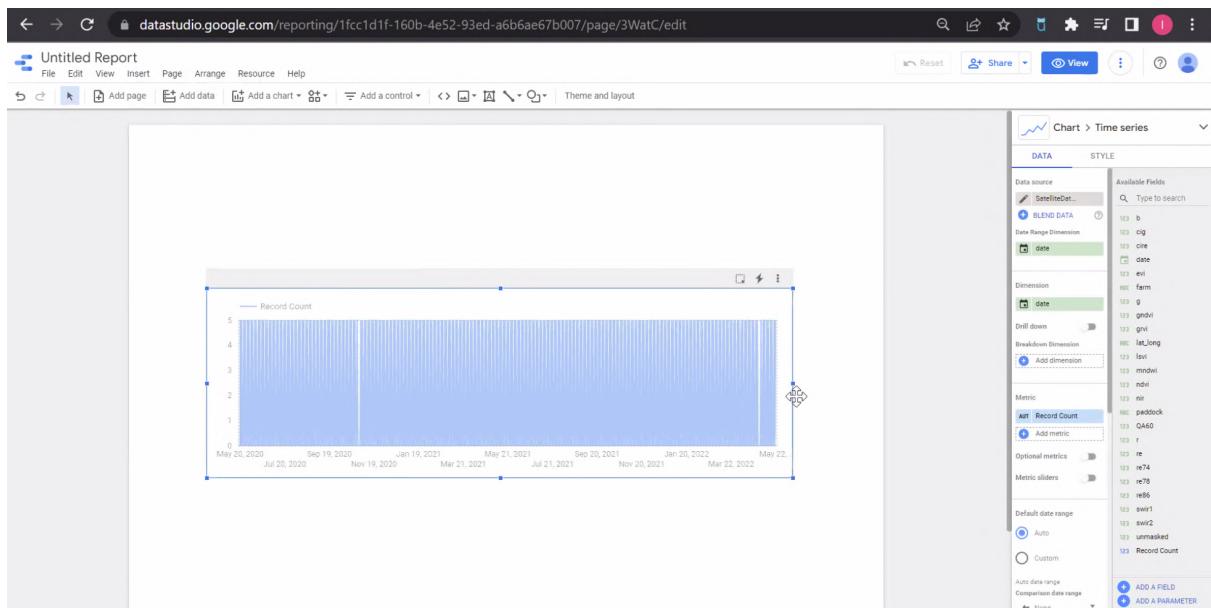
- Next, Select “add to report”:



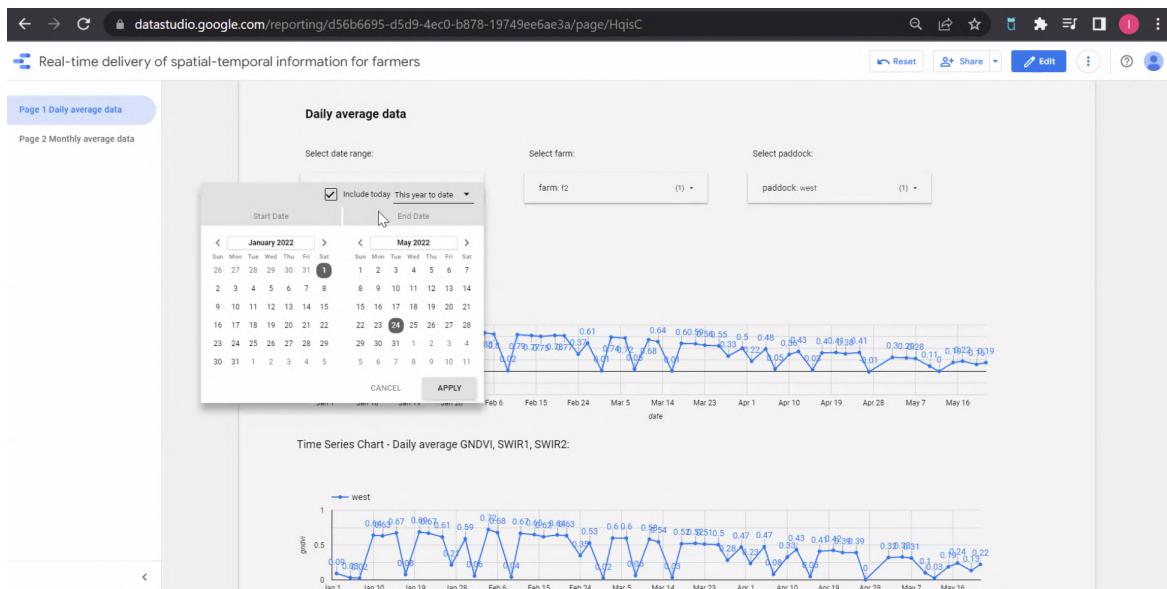
- The user has the option to pick different kinds of visual data representation formats (graphs, bubble map, etc) and can add the relevant fields of interest into the data representations within the report:



- The fields to add are located in the list to the right of the drop down screen of the selected data representation:



- Users can select the date range, farm or paddock from which they want to visualize data:

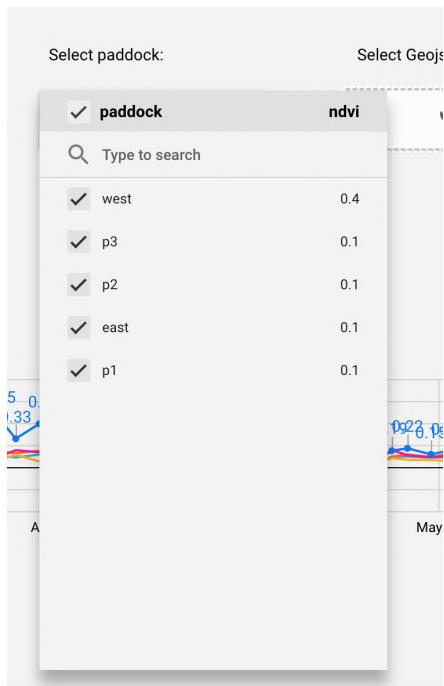


In the finished sample report the options to select are shown below:

Select Date range:

- Select farm:

- Select Paddock:



- The finished sample report is displayed below, the first page shows the “Daily average data” :



- while the second page shows the “Monthly summary data” :

#### Monthly summary data

Select date range:

Jan 1, 2019 - Jun 12, 2022

Select farm:

farm

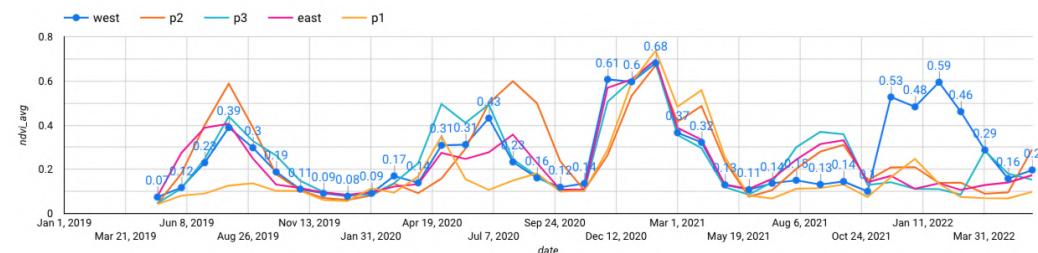
Select paddock:

paddock

Select Geojson file:

geojson\_filename

Time Series Chart - Monthly average NDVI:



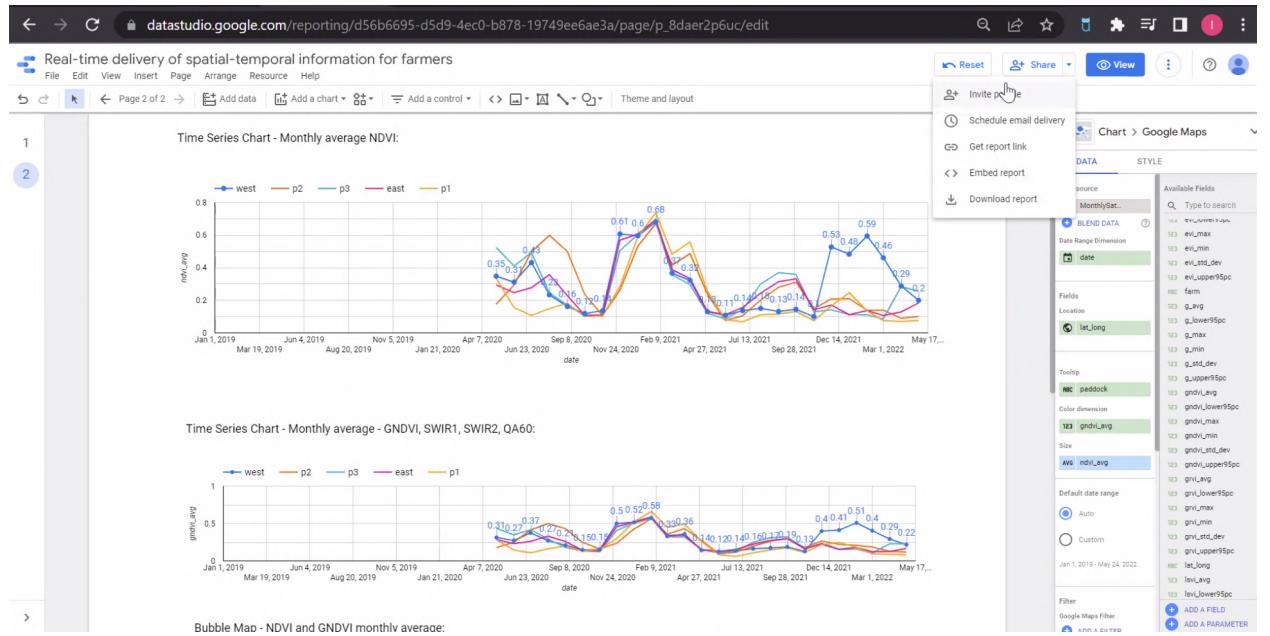
Time Series Chart - Monthly average - GNDVI, SWIR1, SWIR2, QA60:



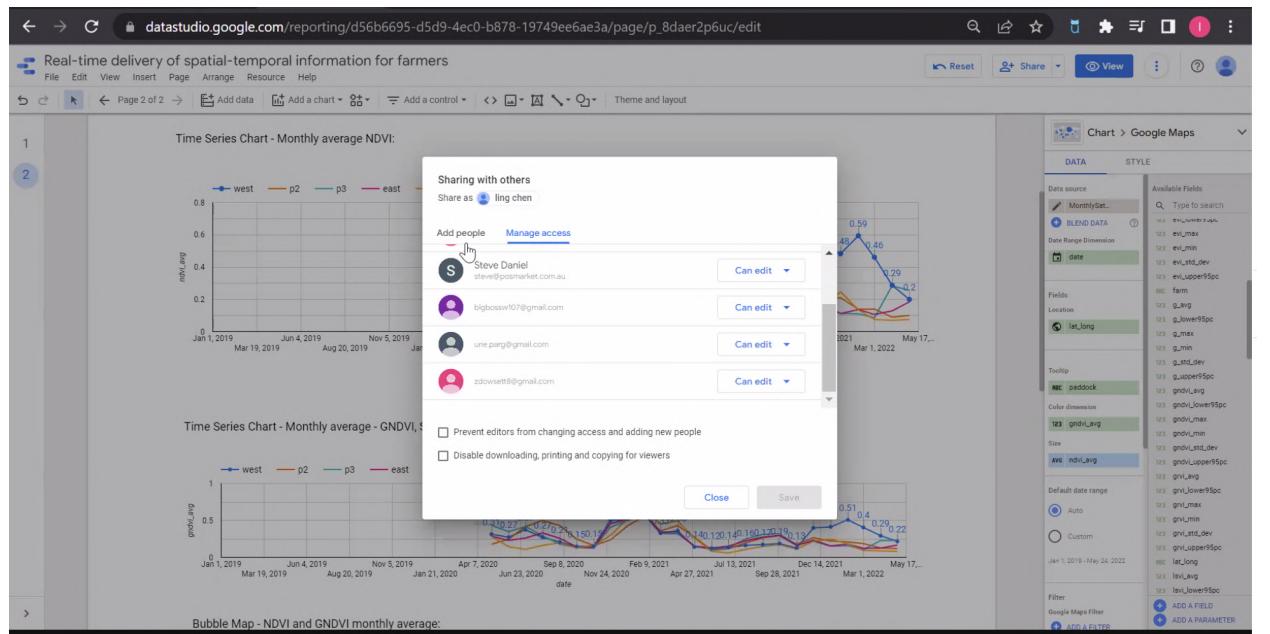
Bubble Map - NDVI and GNDVI monthly average:



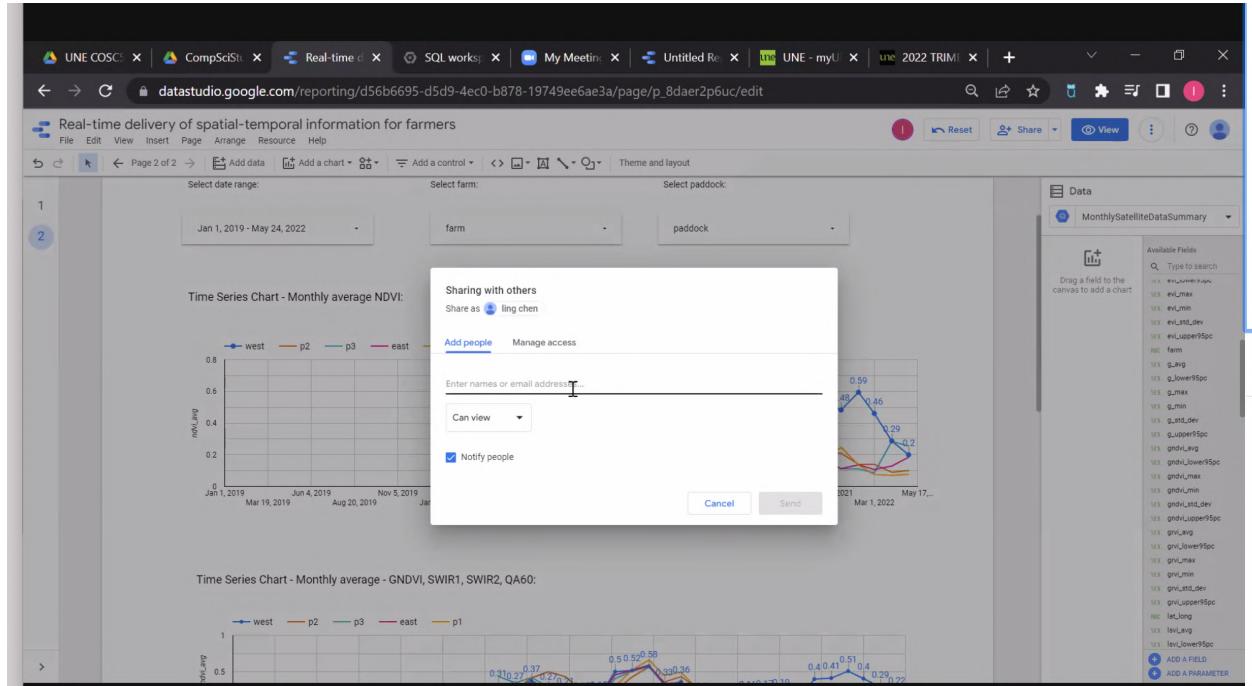
- To share the report, navigate to the top right of the page and click on the “Share+” tab button:



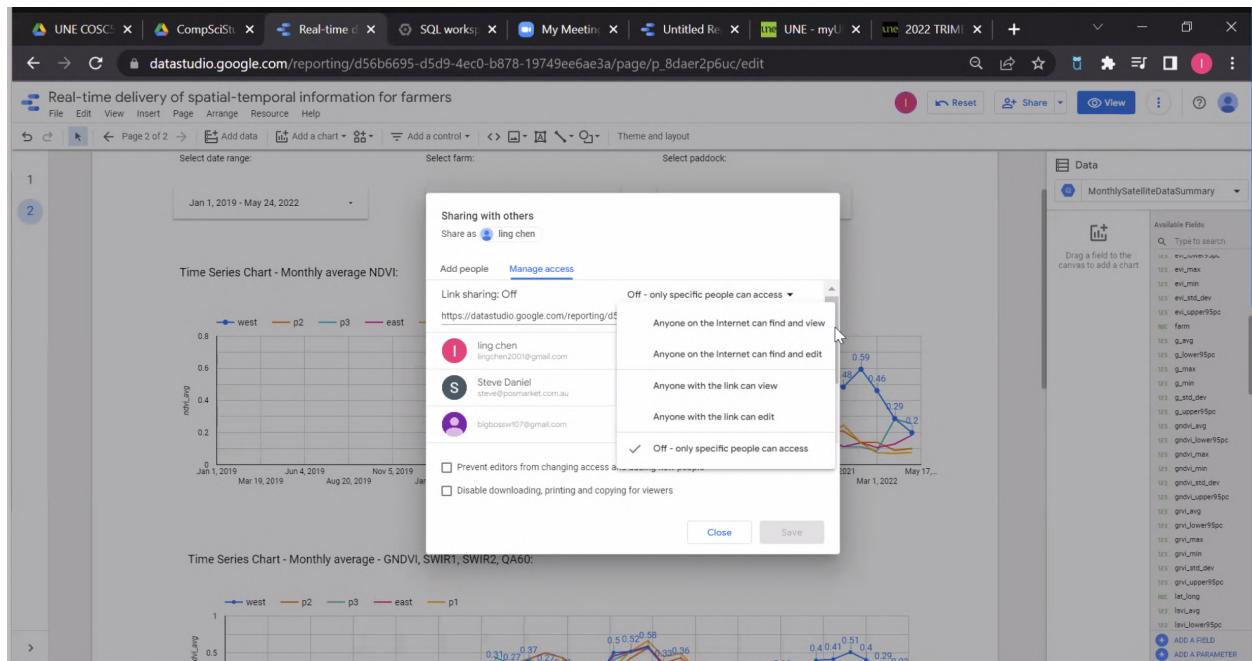
- The user can share the report by sending out an invitation for people to view the report:



- User has the option to add people to share with:



- The authorised user or owner of the report, can manage the access privileges that determine the permissions to view the report and the permissions to edit the report:



- The User is also able to generate a sharable link with the desired permissions to share publicly or with specific people, based on the intended scope of privacy:



- **Please note that:** all sharing, permissions, and management of access to the report can be accomplished via the “Share +” tab drop-down button located at the top right of the page.

This concludes the Installation and User Manual for CropWatch-RDSIF.

To visit the open source github repository established by the author of this Installation and User Manual document, go to:

<https://github.com/Jase-The-Ace/CropWatch-RDSIF>

## **NOTICES**

Copyright 2022 Jason O. Aboh, Steve Daniel, Zoe Dowsett, Qiaoling (Ling) Chen  
Copyright 2022 James Brinkhoff

Licensed under the Apache License, Version 2.0 (the "License");  
you may not use this file except in compliance with the License.

You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software  
distributed under the License is distributed on an "AS IS" BASIS,  
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
See the License for the specific language governing permissions and  
limitations under the License.

## **LICENSES**

Apache License  
Version 2.0, January 2004  
<http://www.apache.org/licenses/>

### **TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION**

#### **1. Definitions.**

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work

(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their

Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

- (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and
- (b) You must cause any modified files to carry prominent notices stating that You changed the files; and
- (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
- (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions.

Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

## END OF TERMS AND CONDITIONS

### APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]"  
replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");  
you may not use this file except in compliance with the License.  
You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.